Nama : Farid Aziz Wicaksono

Kelas : TI-1C

Absen : 14

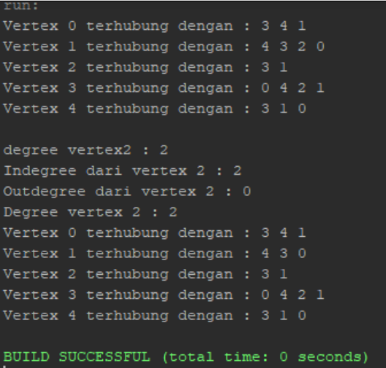
1. Praktikum
2. Praktikum 1

|  |  |
| --- | --- |
| No | Node.java |
| 1  2  3  4  5  6  7  8  9  10  11  12 | package Jobsheet15;  public class Node {  int data;  Node next;  public Node(int data, Node next) {  this.data = data;  this.next = next;  }  } |

|  |  |
| --- | --- |
| No | LinkedList.java |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  112  113  114  115  116  117  118  119  120  121  122  123  124  125  126 | package jobsheet15;  public class LinkedList {  Node head;  int size;  public LinkedList() {  head = null;  size = 0;  }  public boolean isEmpty() {  return head == null;  }  public void addFirst(int item) {  head = new Node(item, head);  size++;  }  public void add(int item, int index) throws Exception {  if (index < 0 || index > size) {  throw new Exception("Nilai index di luar batas");  }  if (isEmpty() || index == 0) {  addFirst(item);  } else {  Node tmp = head;  for (int i = 1; i < index; i++) {  tmp = tmp.next;  }  Node next = (tmp == null) ? null : tmp.next;  tmp.next = new Node(item, next);  }  size++;  }  public void addLast(int item) {  if (isEmpty()) {  addFirst(item);  } else {  Node tmp = head;  while (tmp.next != null) {  tmp = tmp.next;  }  tmp.next = new Node(item, null);  }  size++;  }  public int getFirst() throws Exception {  if (isEmpty()) {  throw new Exception("LinkedList Kosong");  }  return head.data;  }  public int getLast() throws Exception {  if (isEmpty()) {  throw new Exception("LinkedList Kosong");  }  Node tmp = head;  while (tmp.next != null) {  tmp = tmp.next;  }  return tmp.data;  }  public int get(int index) throws Exception {  if (isEmpty() || index >= size) {  throw new Exception("Nilai index di luar batas");  }  Node tmp = head;  for (int i = 0; i < index; i++) {  tmp = tmp.next;  }  return tmp.data;  }  public void remove(int index) throws Exception {  if (isEmpty() || index >= size) {  throw new Exception("NIlai index di luar batas");  }  if (index == 0) {  removeFirst();  } else {  Node prev = head;  Node cur = head.next;  for (int i = 1; i < index; i++) {  prev = cur;  cur = prev.next;  }  prev.next = cur.next;  size--;  }  }  public void removeFirst() {  head = head.next;  size--;  }  public void clear() {  head = null;  size = 0;  }  public void print() {  if (!isEmpty()) {  Node tmp = head;  while (tmp != null) {  System.out.print(tmp.data + "\t");  tmp = tmp.next;  }  System.out.println();  } else {  System.out.println("LinkedLink Kosong");  }  }  public int size() {  return size;  }  } |

|  |  |
| --- | --- |
| No | Graph.java |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82 | public class Graph {  int vertex;  LinkedList list[];  public Graph(int vertex) {  this.vertex = vertex;  list = new LinkedList[vertex];  for (int i = 0; i < vertex; i++) {  list[i] = new LinkedList();  }  }  public void addEdge(int source, int destination) {  list[source].addFirst(destination);  list[destination].addFirst(source);  }  public void degree(int source) throws Exception {  System.out.println("degree vertex " + source + " : " + list[source].size());  int k, totalIn = 0, totalOut = 0;  for (int i = 0; i < vertex; i++) {  for (int j = 0; j < list[i].size(); j++) {  if (list[i].get(j) == source) {  ++totalIn;  }  }  for (k = 0; k < list[source].size(); k++) {  list[source].get(k);  }  totalOut = k;  }  System.out.println("Indegree dari vertex " + source + " : " + totalIn);  System.out.println("Outdegree dari vertex " + source + " : " + totalOut);  System.out.println("degree vertex " + source + " : " + (totalIn + totalOut));  }  public void removeEdge(int source, int destination) throws Exception {  for (int i = 0; i < vertex; i++) {  if (i == destination) {  list[source].remove(destination);  }  }  }  public void removeAllEdges() {  for (int i = 0; i < vertex; i++) {  list[i].clear();  }  System.out.println("Graph berhasil dikosongkan");  }  public void printGraph() throws Exception {  for (int i = 0; i < vertex; i++) {  if (list[i].size() > 0) {  System.out.println("Vertex " + i + " terhubung dengan: ");  for (int j = 0; j < list[i].size(); j++) {  System.out.println(list[i].get(j) + " ");  }  System.out.println("");  }  }  System.out.println(" ");  }  }  class graphMain {  public static void main(String[] args) throws Exception {  Graph graph = new Graph(6);  graph.addEdge(0, 1);  graph.addEdge(0, 4);  graph.addEdge(1, 2);  graph.addEdge(1, 3);  graph.addEdge(1, 4);  graph.addEdge(2, 3);  graph.addEdge(3, 4);  graph.addEdge(3, 0);  graph.printGraph();  graph.degree(2);  graph.removeEdge(1, 2);  graph.printGraph();  }  } |

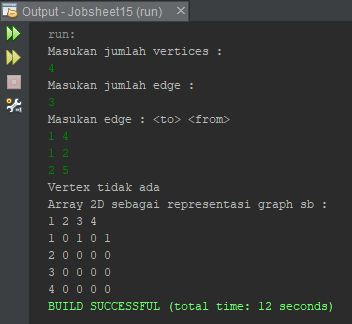
Output :



1. Praktikum 2

|  |  |
| --- | --- |
| No | graphArray.java |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67 | package jobsheet15;  import java.util.\*;  public class graphArray {  private final int vertices;  private int[][] twoD\_array;    public graphArray(int v){  vertices = v;  twoD\_array = new int[vertices + 1][vertices + 1];  }    public void makeEdge(int to, int from, int edge){  try{  twoD\_array[to][from] = edge;  }  catch (ArrayIndexOutOfBoundsException index){  System.out.println("Vertex tidak ada");  }  }  public int getEdge(int to, int from){  try{  return twoD\_array[to][from];  }  catch (ArrayIndexOutOfBoundsException index){  System.out.println("Vertex tidak ada");  }  return -1;  }    public static void main(String[] args) {  int v, e, count = 1, to = 0, from = 0;  Scanner sc = new Scanner(System.in);  Scanner sa = new Scanner(System.in);  graphArray graph;  try {  System.out.println("Masukan jumlah vertices : ");  v = sc.nextInt();  System.out.println("Masukan jumlah edge : ");  e = sa.nextInt();  graph = new graphArray(v);  System.out.println("Masukan edge : <to> <from>");  while (count <= e) {  to = sc.nextInt();  from = sc.nextInt();  graph.makeEdge(to, from, 1);  count++;  }  System.out.println("Array 2D sebagai representasi graph sb : ");  System.out.print("");  for (int i = 1; i <= v; i++) {  System.out.print(i + " ");  }  System.out.println();  for (int i = 1; i <= v; i++) {  System.out.print(i + " ");  for (int j = 1; j <= v; j++) {  System.out.print(graph.getEdge(i, j) + " ");  }  System.out.println();  }  } catch (Exception E) {  System.out.println("Error. Silakan cek kembali");  }  sc.close();  }  } |

Output :



1. Pertanyaan
2. Mengapa graph diimplementasikan dengan double linked list, bukan single linked

list?

Jawab :

karena graph membutuhkan sebuah node yang digunakan untuk indegree dan

outdegree.

1. Jelaskan masing-masingpada dua jenis looping yang terdapat dalam metod

printgraph()!

Jawab :

looping yang pertama digunakan untuk menampilkan vertex ke berapa looping yang kedua digunakan untuk mencari nomor yang terhubung ke vertex

1. Apakah perbedaan graph dengan binary tree pada implementasi nya menggunakan

linked list?

Jawab :

jika dalam binary tree, node yang terkecil akan diletakkan di left, sedangkan node terbesar dari parent akan diletakkan di bagian kanan jika dalam graph, vertex akan diletakkan di bagian kiri, sedangakan di bagian kanan akan memasukkan lintasan dari vertex.

1. Jelaskan dengan contoh perbedaan antara edge dan path pada graph!

Jawab :

edge adalah lintasan penghubung antar simpul dalam graph, sedangkan path adalah urutan dari lintasan ( edge ). Contohnya lintasan j ke p, yang dapat kita sebut sebagai lintasan ( path ) jbcp adalah lintasan ( path ) dari simpul j ke p.

1. Sebutkan beberapa contoh (minimal 3) implementasi graph dalam permasalahan yang

membutuhkan representasi internal dalam memori komputer untuk suatu struktur data!

Jawab :

digunakan untuk pembuatan website, digunakan untuk mendesain web dalam mengalokasi memori computer, pengaplikasian dalam membuat suatu program oleh

para programmer

1. Sebutkan beberapa jenis (minimal 3) algoritma yang menggunakan dasar graph, dan

apakah kegunaan algoritma-algoritma tersebut?

Jawab :

1. Algoritma djikstra merupakan salah satu varian bentuk algoritma popular dalam

pemecahan persoalan terkait masalah optimasi pencarian lintasan terpendek sebuah

lintasan yang mempunyai panjang minimum dari verteks a ke z dalam graph berbobot, bobot tersebut adalah bilangan positif jadi tidak dapat dilalui oleh node

negative

1. Algoritma greedy merupakan jenis algoritma yang menggunakan pendekatan

penyelesaian masalah dengan mencari nilai maksimum sementara pada setiap

langkahnya

1. Algoritma d’satur merupakan algoritma pewarnaan terurut dari derajat simpul tertinggi hingga derajat terendah. Derajat saturasi suatu simpul vi adalah derajat

simpul vi dikurangi banyaknya warna berbeda yang sudah muncul pada tetangga vi

1. Pada class graph terdapat array bertipe linkedlist, yaitu linkedlist list[]. Apakah tujuan

pembuatan variabel tersebut?

Jawab :

array digunakan untuk vertex

1. Apakah perbedaan degree/derajat pada directed dan undirected graph?

Jawab :

undirected : graf yang semua edge miliknya bersifat bi-directional. Dengan kata lain edge, tersebut tidak memiliki arah tertentu. Directed: directed graph adalah graf yang semua edge miliknya bersifat uni-directional, dengan kata lain edge tersebut menuju ke arah tertentu.

1. apakah alasan pemanggilan method addfirst() untuk menambahkan data, bukan

method add jenis lain pada linked list ketika digunakan pada method addedge pada

class graph?

Jawab :

karena pada graph tidak memasukkan index, jadi mengunakan addfirst().

1. Bagaimana cara mendeteksi prev pointer pada saat akan melakukan penghapusan

suatu edge pada graph?

Jawab :

akan dilakukan proses looping jika sudah sesuai maka akan dilakukan proses

remove

1. kenapa uji coba praktikum bagian 1 poin 12 untuk menghapus path yang bukan

merupakan lintasan pertama kali menghasilkan output error atau salah?bagaimana

solusinya?

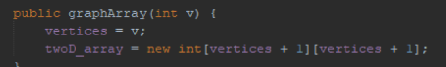
Jawab :



1. Pada implementasi graph menggunakan adjecency matriks. Kenapa jumlah vertices harus ditambahkan dengan 1 pada indeks array

Jawab :

Karena vertex dimasukkan dalam matrix sedangkan indeks pada matrix dimulai dari angka 0, jadi supaya hasil vertex sama dengan yang aslinya maka harus di +1.



1. Apakah kegunaan method getEdge(); ?

Jawab :

Untuk menampilkan suatu lintasan diperlukan pembuatan method getEdge().

1. Termasuk jenis graph apakah uji coba pada praktikum bagian 2?

Jawab :

Uji coba graph bagian 2 menggunakan array 2 dimensi ( Adjacency Matrix ) sebagai representasi graph.

1. Tugas
2. Ubahlah lintasan pada praktikum bagian 1 menjadi inputan!
3. Tambahkan method graphType dengan tipe boolean yang akan membedakan graph

termasuk directed atau undirected graph. Kemudian update seluruh method yang berelasi

dengan method graphType tersebut (hanya menjalankan statement sesuai dengan jenis

graph) pada praktikum bagian 1

1. Modifikasi class Graph sehingga terdapat pilihan menu pada praktikum

bagian 1:

1) Jenis Graph (directed/undirected)

2) Input jumlah vertex

3) addEdge

4) removeEdge

5) removeAllEdges

6) Keluar

Setiap perubahan sesuai dengan menu yang dipilih otomatis akan mencetak kondisi graph

ter-update

1. Modifikasi method removeEdge() pada praktikum bagian 1 agar tidak menghasilkan output yang salah untuk path selain path pertama kali!
2. Buatlah class untuk membuat, menghapus dan menampilkan weighted graph menggunakan representasi adjacency list!
3. Ubahlah tipe data vertex pada seluruh graph pada praktikum bagian 1 dan 2 dari Integer

menjadi String. Misalkan saja setiap vertex yang pada awalnya berupa angka 0,1,2,3, dst.

Akan di rubah menjadi suatu nama daerah Malang, Surabaya, Gresik, Bandung, dst.